# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# SUPPLEMENTAL TO THESIS (3 OF 10: TRAC-MONTEREY 17A SURVEY ANALYSIS (FACTOR AND REGRESSION TREE ANALYSIS))

**RECRUITING THE CYBER LEADER: AN EVALUATION OF THE HUMAN RESOURCE MODEL USED FOR RECRUITING THE ARMY'S "CYBER OPERATIONS OFFICER"**

by

Wallace C. Nicholson
Sean A. Gibbs

September 2017

Thesis Co-Advisors: Steve Mullins
Alejandro Hernandez
Second Reader: William Hatch

**Approved for public release. Distribution is unlimited.**

# 17A Survey Work

*Shingleton*

*August 2, 2017*

I wrote this function to find out how many factors we need, depending on the data set. It is called a 'scree' plot and can graphically tell you the optimal number of factors using the eigenvalues of the data's correlation matrix.

```
myscree<-function(indata){
ev <- eigen(cor(indata)) # get eigenvalues
ap <- parallel(subject=nrow(indata),var=ncol(indata),
               rep=100,cent=.05)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
plotnScree(nS)
}
```

Now, we need to pull in the data. Make sure you put the data somewhere you can reach. For R, the data has to be in .csv format.

```
data1<-read.csv("C:/Users/jsshingl1/Documents/Projects/Survey 17A/Raw Survey Data.csv")
```

First, we are going to look at all of the data without subseting. Some explatory data analysis is a good start.

First, we need to know how many of each rank we have:

```
numRanks<-data1%>%
  group_by(Rank)%>%
  tally()
numRanks$PercRanks<-numRanks$n/sum(numRanks$n)
numRanks
```

```
## # A tibble: 6 x 3
##    Rank     n  PercRanks
##   <int> <int>      <dbl>
## 1     1    17 0.08854167
## 2     2    22 0.11458333
## 3     3    70 0.36458333
## 4     4    42 0.21875000
## 5     5    32 0.16666667
## 6     6     9 0.04687500
```

Thus, most of your observastions came from CPTS.

We are going to look at this for all of the different catagories. What is the mean and std deviation for each of the questions by rank? Sex?

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_20A = mean(X20A), var_20A=var(X20A))
```

```
## # A tibble: 6 x 3
##    Rank mean_20A   var_20A
##   <int>    <dbl>     <dbl>
## 1     1 1.647059 4.2426471
## 2     2 3.500000 2.1666667
## 3     3 3.614286 1.4287785
```

```
## 4     4 3.857143 1.5400697
## 5     5 3.656250 1.7812500
## 6     6 3.888889 0.8611111
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_20B = mean(X20B), var_20B=var(X20B))
```

```
## # A tibble: 6 x 3
##    Rank mean_20B   var_20B
##   <int>    <dbl>     <dbl>
## 1     1 2.058824 4.9338235
## 2     2 3.954545 2.6168831
## 3     3 4.057143 2.2865424
## 4     4 4.214286 2.1236934
## 5     5 4.343750 2.0393145
## 6     6 4.888889 0.1111111
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_20C = mean(X20C), var_20B=var(X20C))
```

```
## # A tibble: 6 x 3
##    Rank mean_20C  var_20B
##   <int>    <dbl>    <dbl>
## 1     1 1.882353 3.860294
## 2     2 3.363636 1.766234
## 3     3 3.357143 2.319876
## 4     4 3.452381 1.765970
## 5     5 3.343750 4.168347
## 6     6 4.000000 1.750000
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_21A = mean(X21A), var_21A=var(X21A))
```

```
## # A tibble: 6 x 3
##    Rank mean_21A  var_21A
##   <int>    <dbl>    <dbl>
## 1     1 2.764706 3.816176
## 2     2 2.818182 3.489177
## 3     3 2.728571 3.128157
## 4     4 3.666667 2.471545
## 5     5 3.531250 2.321573
## 6     6 4.111111 1.111111
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_21B = mean(X21B), var_21B=var(X21B))
```

```
## # A tibble: 6 x 3
##    Rank mean_21B   var_21B
##   <int>    <dbl>     <dbl>
## 1     1 3.882353 1.6102941
## 2     2 3.545455 3.4025974
## 3     3 3.728571 3.3600414
## 4     4 4.095238 1.0150987
```

```
## 5     5 4.093750 1.3135081
## 6     6 4.444444 0.5277778
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_21C = mean(X21C), var_21C=var(X21C))
```

```
## # A tibble: 6 x 3
##    Rank mean_21C   var_21C
##   <int>    <dbl>     <dbl>
## 1     1 3.941176 0.8088235
## 2     2 4.000000 2.1904762
## 3     3 4.114286 1.6099379
## 4     4 4.166667 0.7764228
## 5     5 3.937500 1.3508065
## 6     6 4.333333 0.7500000
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_36A = mean(X36A), var_36A=var(X36A))
```

```
## # A tibble: 6 x 3
##    Rank mean_36A   var_36A
##   <int>    <dbl>     <dbl>
## 1     1 3.882353 2.4852941
## 2     2 4.090909 2.1818182
## 3     3 4.642857 0.5807453
## 4     4 4.880952 0.1562137
## 5     5 4.781250 0.1764113
## 6     6 5.000000 0.0000000
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_36B = mean(X36B), var_36B=var(X36B))
```

```
## # A tibble: 6 x 3
##    Rank mean_36B   var_36B
##   <int>    <dbl>     <dbl>
## 1     1 3.823529 2.6544118
## 2     2 4.045455 2.0454545
## 3     3 4.314286 1.0012422
## 4     4 4.119048 1.1806039
## 5     5 4.156250 0.8457661
## 6     6 4.333333 0.7500000
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_36C = mean(X36C), var_36C=var(X36C))
```

```
## # A tibble: 6 x 3
##    Rank mean_36C  var_36C
##   <int>    <dbl>    <dbl>
## 1     1 3.294118 2.845588
## 2     2 4.045455 2.140693
## 3     3 3.285714 2.380952
## 4     4 3.547619 1.619628
## 5     5 3.000000 2.322581
```

```
## 6     6 3.333333 1.750000
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_36D = mean(X36D), var_36D=var(X36D))
```

```
## # A tibble: 6 x 3
##    Rank  mean_36D   var_36D
##   <int>     <dbl>     <dbl>
## 1     1 0.4705882 0.8897059
## 2     2 2.6363636 4.2424242
## 3     3 3.1714286 1.9412008
## 4     4 2.5476190 1.8147503
## 5     5 2.5625000 1.2862903
## 6     6 2.3333333 2.0000000
```

```
data1 %>%
  group_by(Rank) %>%
  summarise (mean_36E = mean(X36E), var_36E=var(X36E))
```

```
## # A tibble: 6 x 3
##    Rank mean_36E  var_36E
##   <int>    <dbl>    <dbl>
## 1     1 3.411765 3.507353
## 2     2 3.500000 2.642857
## 3     3 3.228571 1.860041
## 4     4 3.047619 1.753775
## 5     5 3.375000 1.725806
## 6     6 3.555556 2.277778
```

Now for some correlation plots. I found a nice package that make some really nice ones. I am also going to include Rank and

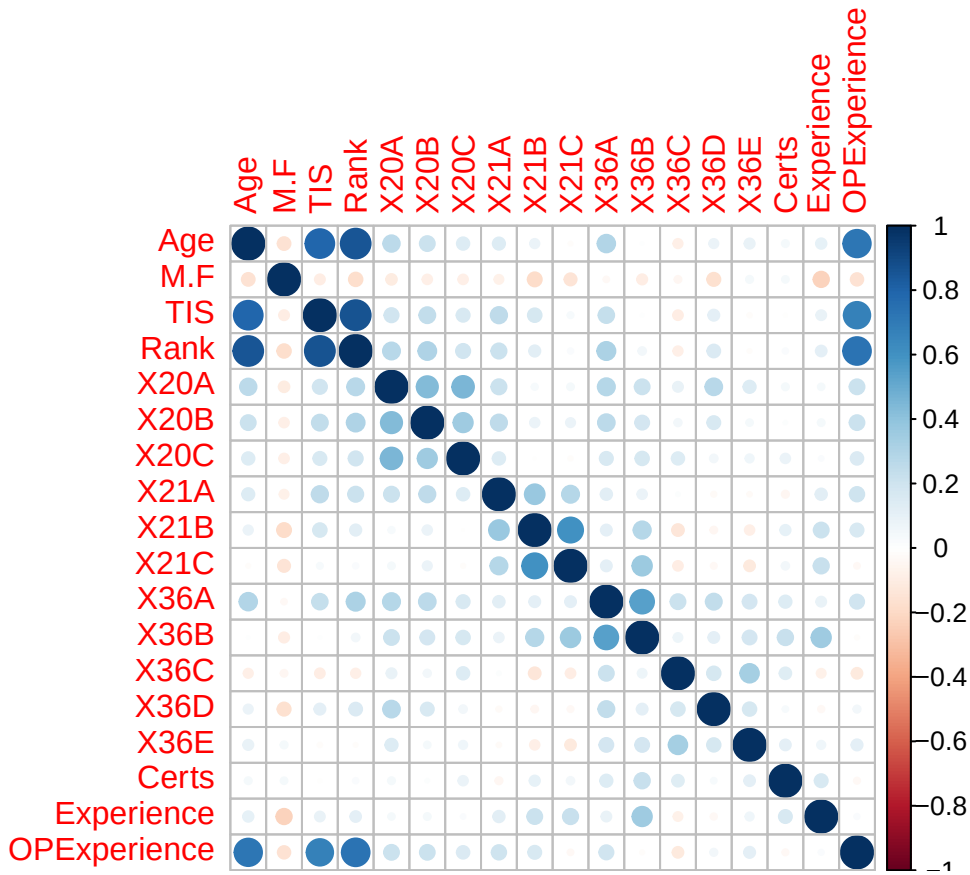
```
data2<-data1[,c(1:4,10:15, 26:32, 36)]
cor(data1[,c(1:4,10:15, 26:32, 36)])
```

```
##                       Age         M.F          TIS        Rank        X20A
## Age           1.000000000 -0.15813047  0.794763738  0.85251319  0.26050074
## M.F          -0.158130474  1.00000000 -0.095725760 -0.17623467 -0.10928884
## TIS           0.794763738 -0.09572576  1.000000000  0.86052097  0.19320539
## Rank          0.852513195 -0.17623467  0.860520972  1.00000000  0.27827924
## X20A          0.260500739 -0.10928884  0.193205393  0.27827924  1.00000000
## X20B          0.213550011 -0.08904433  0.243750719  0.30580742  0.43199780
## X20C          0.148423328 -0.08968785  0.165117153  0.19215268  0.45079949
## X21A          0.144073644 -0.07781946  0.255941275  0.21989852  0.21805909
## X21B          0.082672547 -0.18837050  0.178993901  0.12112428  0.03857154
## X21C         -0.015069811 -0.14047401  0.035416195  0.02857091  0.04545956
## X36A          0.296917396 -0.03105079  0.234235631  0.31396112  0.28109536
## X36B          0.006779977 -0.09878851  0.007288116  0.05786948  0.21386223
## X36C         -0.081686868 -0.04599898 -0.092430880 -0.08465124  0.09670597
## X36D          0.081038886 -0.16528379  0.117962538  0.15817929  0.27128354
## X36E          0.099279320  0.04931653 -0.019609407 -0.01514267  0.14516250
## Certs         0.047423987  0.04420725 -0.009327496  0.02905380  0.04783046
## Experience    0.105577724 -0.22669421  0.094456855  0.11152387  0.04244444
## OPExperience  0.722818211 -0.15000959  0.674862392  0.73890319  0.21369219
##                     X20B         X20C        X21A         X21B        X21C
## Age           0.21355001  0.148423328  0.14407364  0.082672547 -0.01506981
```

```
## M.F          -0.08904433 -0.089687848 -0.07781946 -0.188370500 -0.14047401
## TIS           0.24375072  0.165117153  0.25594127  0.178993901  0.03541620
## Rank          0.30580742  0.192152678  0.21989852  0.121124278  0.02857091
## X20A          0.43199780  0.450799490  0.21805909  0.038571540  0.04545956
## X20B          1.00000000  0.356852097  0.25131985  0.086679453  0.08057245
## X20C          0.35685210  1.000000000  0.14957967 -0.003160218 -0.01706172
## X21A          0.25131985  0.149579669  1.00000000  0.376037874  0.28096068
## X21B          0.08667945 -0.003160218  0.37603787  1.000000000  0.60323993
## X21C          0.08057245 -0.017061717  0.28096068  0.603239928  1.00000000
## X36A          0.26943699  0.164836581  0.12650176  0.115997774  0.11502175
## X36B          0.18806595  0.176396625  0.08954746  0.286394819  0.36424600
## X36C          0.05278756  0.149068698  0.01462158 -0.134900362 -0.09119579
## X36D          0.16249718  0.058131991 -0.02628114 -0.045061505 -0.03467009
## X36E          0.04350882  0.065711354 -0.02060765 -0.087234519 -0.11236593
## Certs         0.01325490  0.088070445 -0.04894662  0.101989816  0.05541870
## Experience    0.04651581  0.021831830  0.12595517  0.214456996  0.22394724
## OPExperience  0.21124985  0.152887535  0.20045565  0.163317593 -0.03242305
##                     X36A         X36B        X36C        X36D        X36E
## Age           0.29691740  0.006779977 -0.08168687  0.08103889  0.09927932
## M.F          -0.03105079 -0.098788510 -0.04599898 -0.16528379  0.04931653
## TIS           0.23423563  0.007288116 -0.09243088  0.11796254 -0.01960941
## Rank          0.31396112  0.057869480 -0.08465124  0.15817929 -0.01514267
## X20A          0.28109536  0.213862231  0.09670597  0.27128354  0.14516250
## X20B          0.26943699  0.188065951  0.05278756  0.16249718  0.04350882
## X20C          0.16483658  0.176396625  0.14906870  0.05813199  0.06571135
## X21A          0.12650176  0.089547456  0.01462158 -0.02628114 -0.02060765
## X21B          0.11599777  0.286394819 -0.13490036 -0.04506151 -0.08723452
## X21C          0.11502175  0.364245996 -0.09119579 -0.03467009 -0.11236593
## X36A          1.00000000  0.540426800  0.21381082  0.24683256  0.18566258
## X36B          0.54042680  1.000000000  0.07627077  0.11471948  0.18660407
## X36C          0.21381082  0.076270773  1.00000000  0.17753609  0.33183250
## X36D          0.24683256  0.114719478  0.17753609  1.00000000  0.17476232
## X36E          0.18566258  0.186604071  0.33183250  0.17476232  1.00000000
## Certs         0.14547254  0.228449367  0.13183525  0.03180425  0.11244215
## Experience    0.09484138  0.359717623 -0.07676999 -0.03872512  0.06596468
## OPExperience  0.19225953 -0.016870503 -0.11233175  0.05030855  0.11843457
##                     Certs  Experience OPExperience
## Age           0.047423987  0.10557772   0.72281821
## M.F           0.044207250 -0.22669421  -0.15000959
## TIS          -0.009327496  0.09445686   0.67486239
## Rank          0.029053795  0.11152387   0.73890319
## X20A          0.047830457  0.04244444   0.21369219
## X20B          0.013254899  0.04651581   0.21124985
## X20C          0.088070445  0.02183183   0.15288753
## X21A         -0.048946624  0.12595517   0.20045565
## X21B          0.101989816  0.21445700   0.16331759
## X21C          0.055418697  0.22394724  -0.03242305
## X36A          0.145472540  0.09484138   0.19225953
## X36B          0.228449367  0.35971762  -0.01687050
## X36C          0.131835247 -0.07676999  -0.11233175
## X36D          0.031804254 -0.03872512   0.05030855
## X36E          0.112442152  0.06596468   0.11843457
## Certs         1.000000000  0.16360131  -0.03649618
## Experience    0.163601309  1.00000000   0.02664683
```

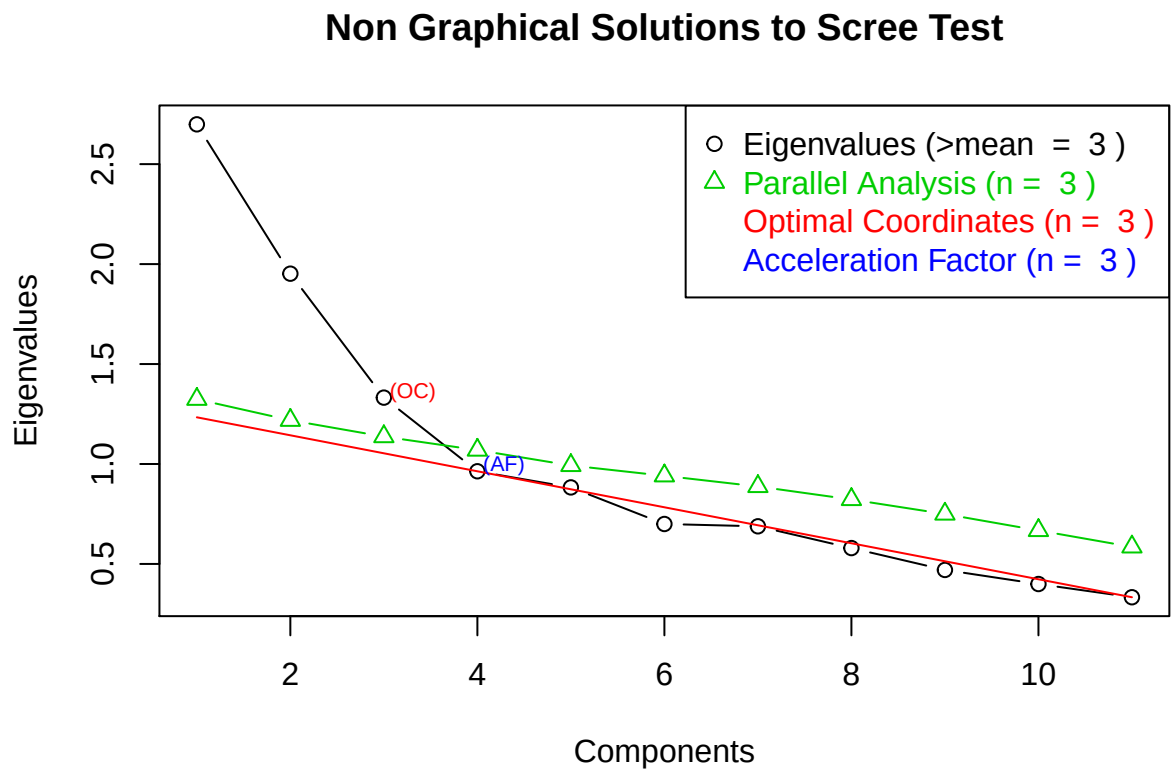
```
## OPExperience -0.036496180  0.02664683   1.00000000
```

```r
corrplot(cor(data1[,c(1:4,10:15, 26:32, 36)]), method="circle")
```




Finally, a little bit of factor analysis

```r
myscree(data1[,c(10:15, 26:30)])
```




This graph says that we should use a three factor model for these 12 questions. Let's give it a try!

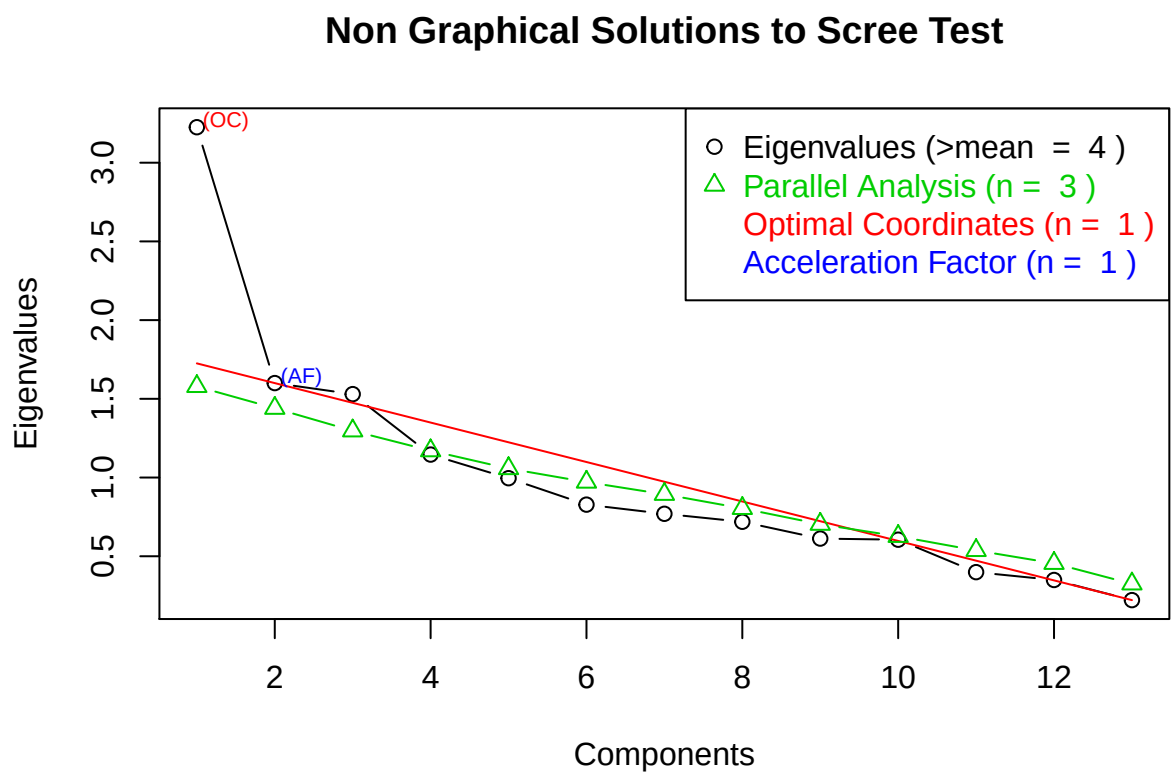
```
fa1<-factanal(data1[,c(10:15, 26:30)], factors=3, rotation="varimax")
fa1
```

```
##
## Call:
## factanal(x = data1[, c(10:15, 26:30)], factors = 3, rotation = "varimax")
##
## Uniquenesses:
##  X20A  X20B  X20C  X21A  X21B  X21C  X36A  X36B  X36C  X36D  X36E
## 0.446 0.631 0.665 0.717 0.369 0.417 0.487 0.375 0.873 0.869 0.848
##
## Loadings:
##      Factor1 Factor2 Factor3
## X20A          0.709   0.224
## X20B  0.109   0.576   0.158
## X20C          0.559   0.152
## X21A  0.401   0.338
## X21B  0.791
## X21C  0.763
## X36A  0.179   0.198   0.664
## X36B  0.431           0.659
## X36C -0.162   0.107   0.299
## X36D          0.217   0.280
## X36E -0.130           0.360
##
```

```
##                 Factor1 Factor2 Factor3
## SS loadings       1.648   1.371   1.284
## Proportion Var    0.150   0.125   0.117
## Cumulative Var    0.150   0.274   0.391
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 41.31 on 25 degrees of freedom.
## The p-value is 0.0213
```

Subsetting now: 17A as Rater. Notice that this allows me to add question 27A and 27B into the factor analysis.

```
subdata1<-data1%>%
  filter(!is.na(X27A))
myscree(subdata1[,c(10:17, 26:30)])
```



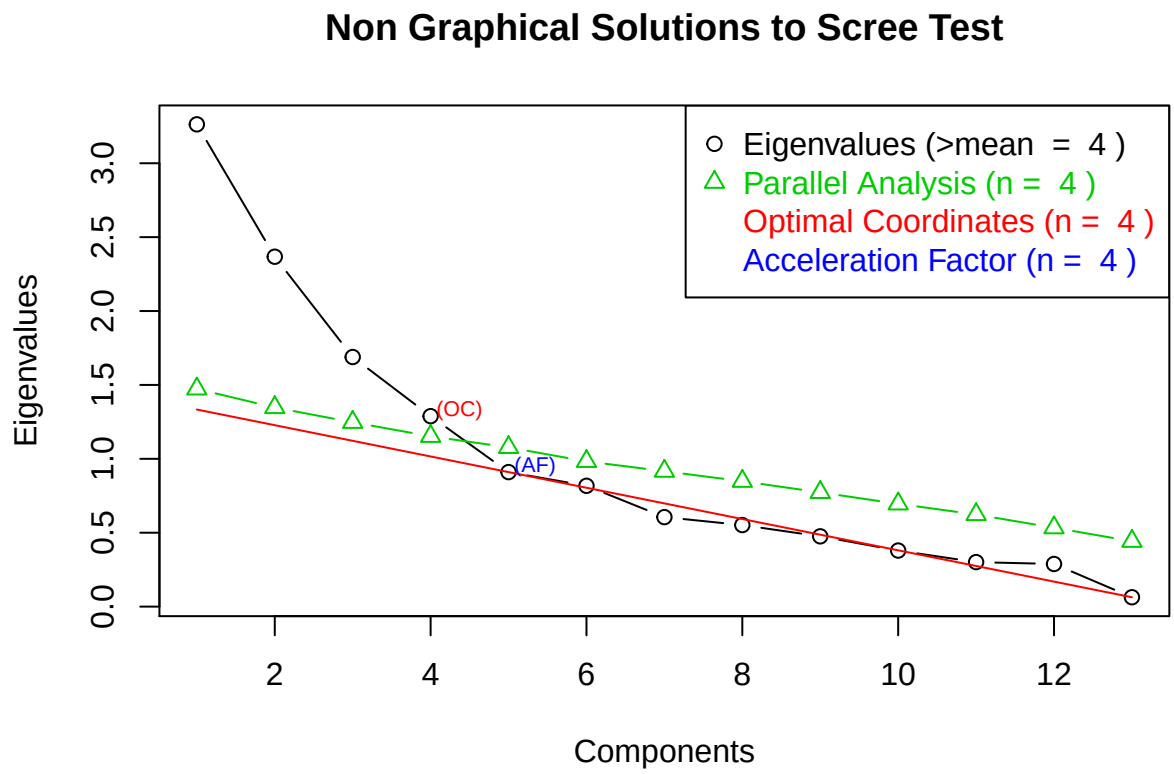

```
##This says 1, but we are going to still use 3.
fa1<-factanal(subdata1[,c(10:17, 26:30)], factors=3, rotation="varimax")
fa1
```

```
##
## Call:
## factanal(x = subdata1[, c(10:17, 26:30)], factors = 3, rotation = "varimax")
##
## Uniquenesses:
##  X20A  X20B  X20C  X21A  X21B  X21C  X27A  X27B  X36A  X36B  X36C  X36D
## 0.752 0.767 0.872 0.826 0.078 0.661 0.105 0.447 0.507 0.443 0.943 0.857
##  X36E
```

```
## 0.815
##
## Loadings:
##      Factor1 Factor2 Factor3
## X20A  0.192   0.456
## X20B  0.102   0.464
## X20C  0.157   0.314
## X21A  0.330           0.249
## X21B          0.152   0.946
## X21C  0.118   0.195   0.536
## X27A  0.927   0.142   0.123
## X27B  0.734   0.118
## X36A          0.694
## X36B  0.176   0.646   0.330
## X36C          0.193  -0.136
## X36D -0.163   0.336
## X36E  0.271   0.246  -0.226
##
##                Factor1 Factor2 Factor3
## SS loadings      1.737   1.729   1.462
## Proportion Var   0.134   0.133   0.112
## Cumulative Var   0.134   0.267   0.379
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 30.39 on 42 degrees of freedom.
## The p-value is 0.909
```

Subsetting now: 17A not Rater. We get to add 28A and 28B.

```
subdata1<-data1%>%
  filter(is.na(X27A))
myscree(subdata1[,c(10:15,18:19,26:30)])
```



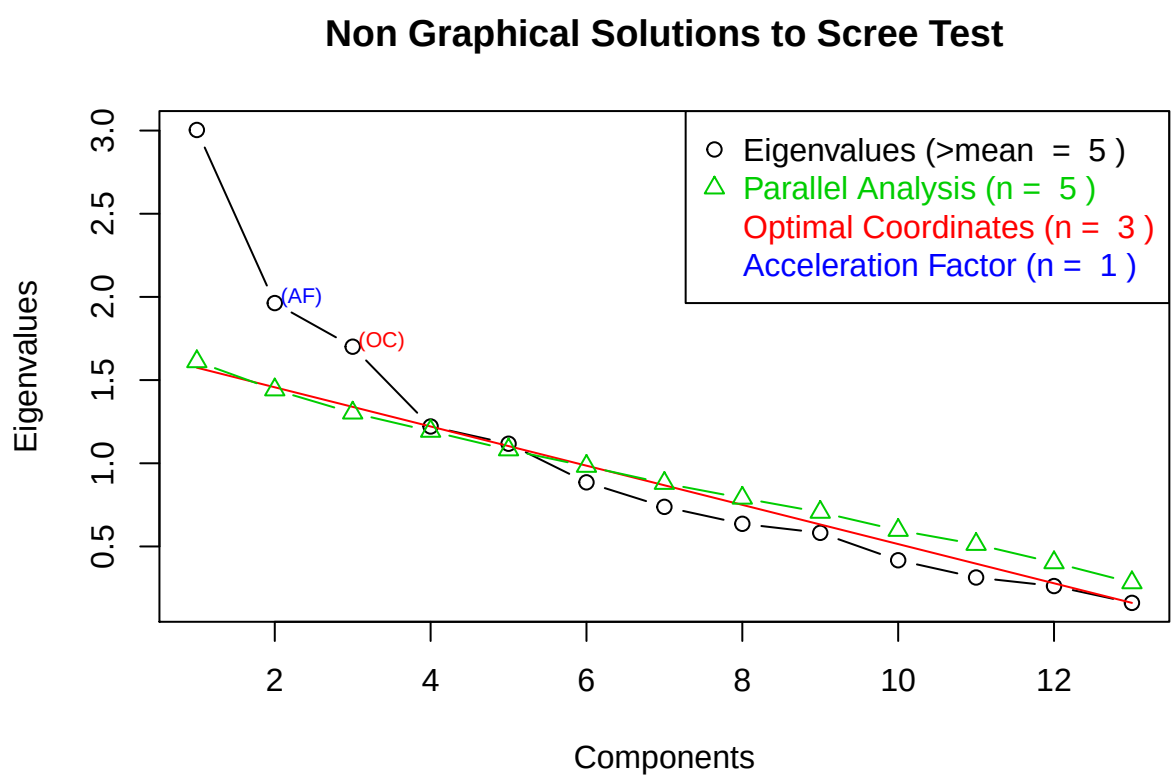

```
##This one says 4, so 4 it is!
fa1<-factanal(subdata1[,c(10:15,18:19,26:30)], factors=4, rotation="varimax")
fa1
```

```
##
## Call:
## factanal(x = subdata1[, c(10:15, 18:19, 26:30)], factors = 4,     rotation = "varimax")
##
## Uniquenesses:
##  X20A  X20B  X20C  X21A  X21B  X21C  X28A  X28B  X36A  X36B  X36C  X36D
## 0.387 0.534 0.516 0.647 0.429 0.237 0.005 0.119 0.442 0.414 0.731 0.817
##  X36E
## 0.785
##
## Loadings:
##      Factor1 Factor2 Factor3 Factor4
## X20A  0.166           0.732   0.223
## X20B  0.107           0.667
## X20C                  0.677   0.132
## X21A  0.361   0.370   0.293
## X21B  0.273   0.686          -0.156
## X21C          0.868
## X28A  0.986   0.121
## X28B  0.912   0.136   0.177
## X36A  0.180   0.256   0.178   0.654
## X36B          0.501           0.576
## X36C         -0.146           0.484
```

```
## X36D         -0.110   0.255   0.325
## X36E         -0.112           0.447
##
##                Factor1 Factor2 Factor3 Factor4
## SS loadings      2.095   1.773   1.670   1.399
## Proportion Var   0.161   0.136   0.128   0.108
## Cumulative Var   0.161   0.298   0.426   0.534
##
## Test of the hypothesis that 4 factors are sufficient.
## The chi square statistic is 52.06 on 32 degrees of freedom.
## The p-value is 0.014
```

Subsetting now: 17A Senior Rater. We get to add 30A and 30B.

```
subdata1<-data1%>%
  filter(!is.na(X30A))
myscree(subdata1[,c(10:15,20:21,26:30)])
```



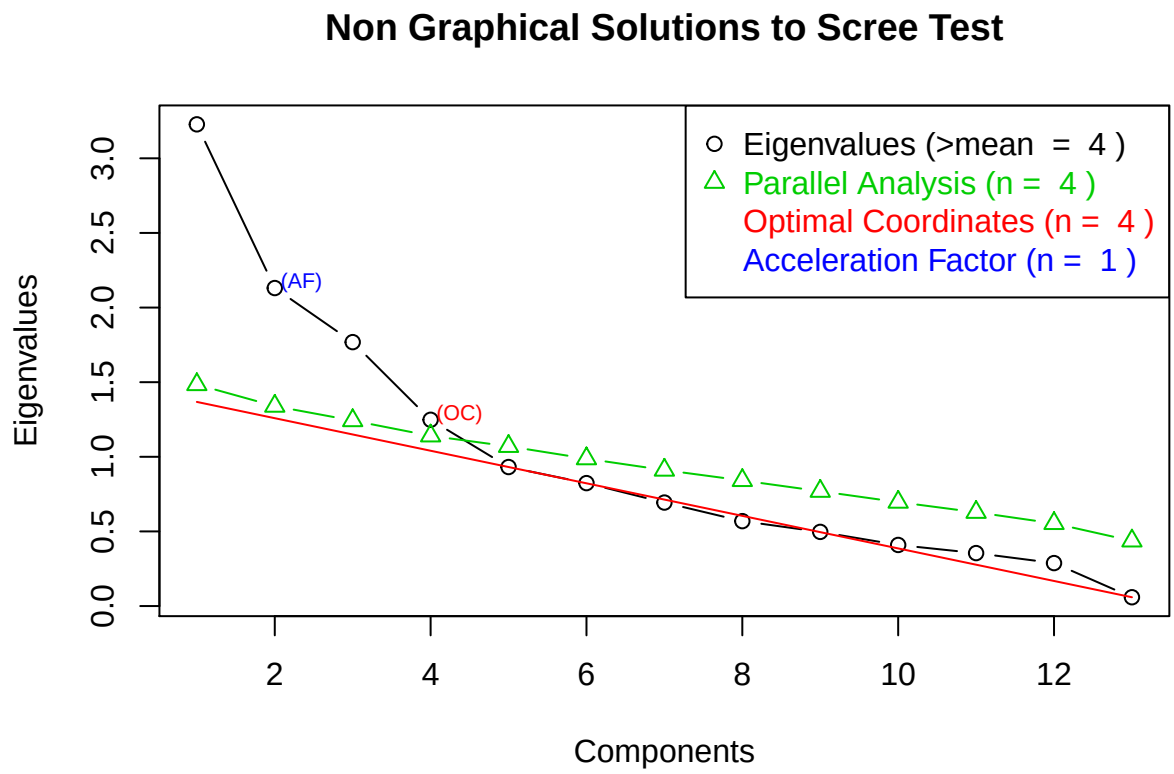

```
##Go with 2. Best p-value
fa1<-factanal(subdata1[,c(10:15,20:21,26:30)], factors=2, rotation="varimax")
fa1
```

```
##
## Call:
## factanal(x = subdata1[, c(10:15, 20:21, 26:30)], factors = 2,     rotation = "varimax")
##
## Uniquenesses:
##  X20A  X20B  X20C  X21A  X21B  X21C  X30A  X30B  X36A  X36B  X36C  X36D
```

```
## 0.753 0.503 0.874 0.983 0.931 0.965 0.005 0.349 0.465 0.486 0.967 0.874
##  X36E
## 0.868
##
## Loadings:
##      Factor1 Factor2
## X20A  0.489
## X20B  0.699
## X20C  0.339   0.106
## X21A  0.101
## X21B  0.221   0.144
## X21C  0.165
## X30A          0.996
## X30B          0.807
## X36A  0.705   0.194
## X36B  0.708   0.112
## X36C  0.147   0.104
## X36D  0.352
## X36E  0.276   0.236
##
##                Factor1 Factor2
## SS loadings      2.152   1.825
## Proportion Var   0.166   0.140
## Cumulative Var   0.166   0.306
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 95.58 on 53 degrees of freedom.
## The p-value is 0.000305
```

Subsetting now: 17A not Senior Rater. We get to add 31A and 31B.

```
subdata1<-data1%>%
  filter(is.na(X30A))
myscree(subdata1[,c(10:15,22:23,26:30)])
```




```
##Try 4. Go with 3.
fa1<-factanal(subdata1[,c(10:15,22:23,26:30)], factors=3, rotation="varimax")
fa1
```

```
##
## Call:
## factanal(x = subdata1[, c(10:15, 22:23, 26:30)], factors = 3,     rotation = "varimax")
##
## Uniquenesses:
##  X20A  X20B  X20C  X21A  X21B  X21C  X31A  X31B  X36A  X36B  X36C  X36D
## 0.329 0.701 0.579 0.662 0.511 0.215 0.058 0.073 0.777 0.617 0.947 0.881
##  X36E
## 0.947
##
## Loadings:
##      Factor1 Factor2 Factor3
## X20A  0.213   0.120   0.782
## X20B  0.206   0.106   0.495
## X20C                  0.640
## X21A  0.400   0.395   0.150
## X21B  0.288   0.613  -0.175
## X21C          0.869  -0.160
## X31A  0.965   0.102
## X31B  0.957
## X36A  0.124   0.342   0.300
## X36B          0.599   0.150
## X36C                  0.220
```

```
## X36D                  0.345
## X36E                  0.212
##
##                Factor1 Factor2 Factor3
## SS loadings       2.21    1.82   1.674
## Proportion Var    0.17    0.14   0.129
## Cumulative Var    0.17    0.31   0.439
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 91.62 on 42 degrees of freedom.
## The p-value is 1.5e-05
```

After relooking at the data, this is my recommendtion:

1) Create a score for each of the factors, using all of the data from the dataset (all 190ish observations)

```
w1sum<-0.401+0.791+0.763
w1.1<-0.401/w1sum
w2.1<-0.791/w1sum
w3.1<-0.763/w1sum
w2sum<-0.709+0.576+0.559
w1.2<-0.709/w2sum
w2.2<-0.576/w2sum
w3.2<-0.559/w2sum
w3sum<-0.664+0.659
w1.3<-0.664/w3sum
w2.3<-0.659/w3sum


data1$FactorScore1<-(w1.1*data1$X21A+w2.1*data1$X21B+w3.1*data1$X21C)
data1$FactorScore2<-(w1.2*data1$X20A+w2.2*data1$X20B+w3.2*data1$X20C)
data1$FactorScore3<-(w1.3*data1$X36A+w2.3*data1$X36B)
```

Now, based on the demographic data, predict the factor score. We can do this by splitting the data into 2 sets, the training set and the test set.

```
traindata1<-sample_n(data1, 0.7*dim(data1)[1])
testdata1<-data1[-as.numeric(rownames(traindata1)),]
```

Now we can do a little regression and prediction with these data sets and see how we did:

```
lmFact1<-lm(traindata1$FactorScore1~traindata1$Rank+traindata1$BS+traindata1$Certs+traindata1$Experienc
```

Get Rid of Rank, No.UG.Degree, and Rank.1, whatever that is.

```
lmFact1<-lm(traindata1$FactorScore1~traindata1$Age+traindata1$M.F+traindata1$TIS+traindata1$TIG)
summary(lmFact1)
```

```
##
## Call:
## lm(formula = traindata1$FactorScore1 ~ traindata1$Age + traindata1$M.F +
##     traindata1$TIS + traindata1$TIG)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5316 -0.4323  0.1966  0.6588  1.8267
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.48501    0.38719  11.583  < 2e-16 ***
## traindata1$Age -0.26460    0.12397  -2.134  0.03470 *
## traindata1$M.F -0.53175    0.25854  -2.057  0.04172 *
## traindata1$TIS  0.27841    0.10280   2.708  0.00768 **
## traindata1$TIG  0.05303    0.07229   0.733  0.46460
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9332 on 129 degrees of freedom
## Multiple R-squared:  0.09246,    Adjusted R-squared:  0.06432
## F-statistic: 3.286 on 4 and 129 DF,  p-value: 0.01334
```

Get rid of TIG. Probably taken care of by TIS

```
lmFact1<-lm(traindata1$FactorScore1~traindata1$Age+traindata1$M.F+traindata1$TIS)
summary(lmFact1)
```

```
##
## Call:
## lm(formula = traindata1$FactorScore1 ~ traindata1$Age + traindata1$M.F +
##     traindata1$TIS)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5779 -0.3973  0.1612  0.6675  1.7721
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.5137     0.3845  11.739  < 2e-16 ***
## traindata1$Age  -0.2604     0.1236  -2.107  0.03707 *
## traindata1$M.F  -0.5377     0.2579  -2.085  0.03906 *
## traindata1$TIS   0.2935     0.1005   2.920  0.00413 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9315 on 130 degrees of freedom
## Multiple R-squared:  0.08868,    Adjusted R-squared:  0.06765
## F-statistic: 4.217 on 3 and 130 DF,  p-value: 0.006992
```

No TIS

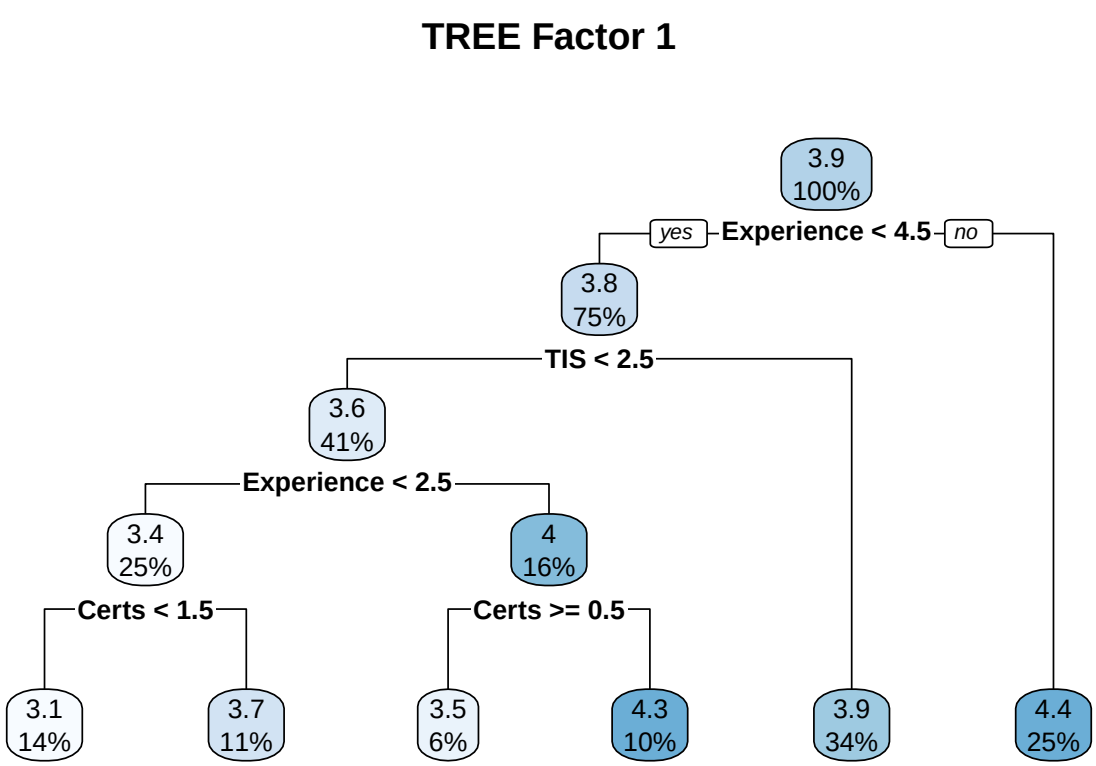
```
lmFact1<-lm(traindata1$FactorScore1~traindata1$Age+traindata1$M.F)
summary(lmFact1)  ##This sucks.
```

```
##
## Call:
## lm(formula = traindata1$FactorScore1 ~ traindata1$Age + traindata1$M.F)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7662 -0.3885  0.1726  0.6782  1.5217
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.36374    0.39187  11.136   <2e-16 ***
```

```
## traindata1$Age  0.03210    0.07447   0.431    0.667
## traindata1$M.F -0.49086    0.26474  -1.854    0.066 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9579 on 131 degrees of freedom
## Multiple R-squared:  0.02891,    Adjusted R-squared:  0.01408
## F-statistic:  1.95 on 2 and 131 DF,  p-value: 0.1464
```

I am going to try a different method.

```
attach(traindata1)
treeFact1<-rpart(FactorScore1~Rank+Age+TIS+BS+Certs+Experience+MS+OPExperience, cp=0.014)
rpart.plot(treeFact1, main="TREE Factor 1")
```



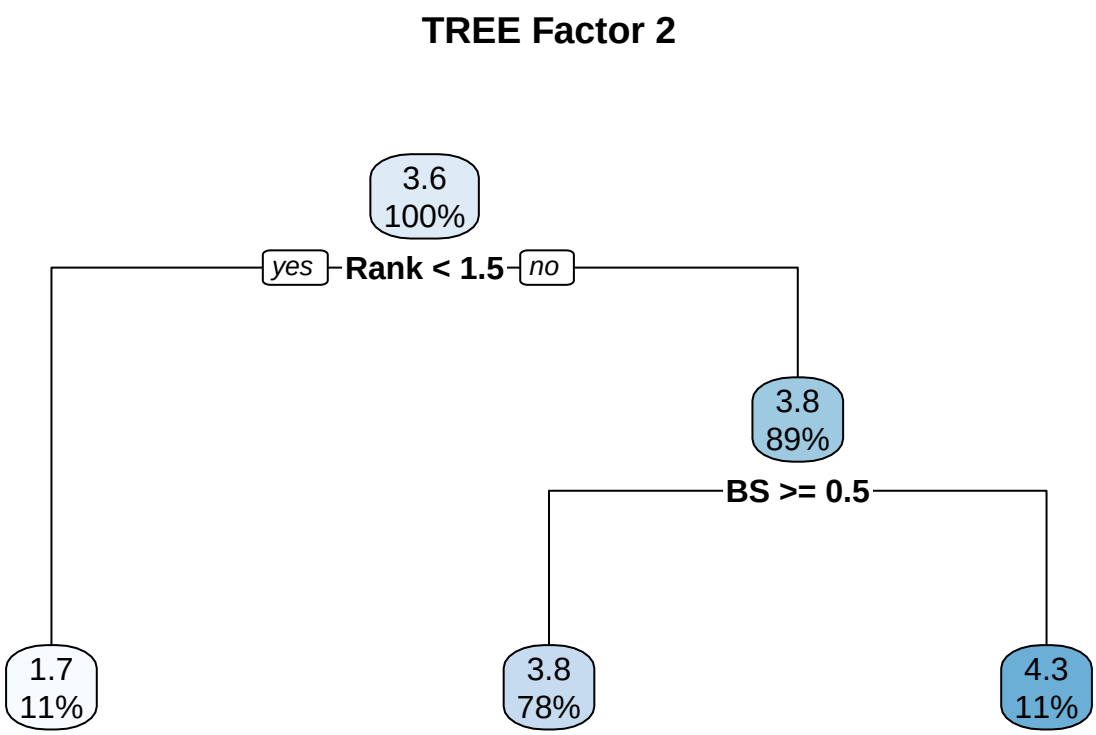

```
# text(treeFact1)

treeFact2<-rpart(FactorScore2~Rank+Age+TIS+BS+Certs+Experience+MS+OPExperience, cp=.014)
printcp(treeFact2)
```

```
##
## Regression tree:
## rpart(formula = FactorScore2 ~ Rank + Age + TIS + BS + Certs +
##     Experience + MS + OPExperience, cp = 0.014)
##
## Variables actually used in tree construction:
## [1] BS   Rank
##
```

```
## Root node error: 222.67/134 = 1.6617
##
## n= 134
##
##         CP nsplit rel error  xerror    xstd
## 1 0.262216      0   1.00000 1.01317 0.16520
## 2 0.016825      1   0.73778 0.77679 0.11935
## 3 0.014000      2   0.72096 0.84338 0.11833
```

```
rpart.plot(treeFact2, main="TREE Factor 2")
```



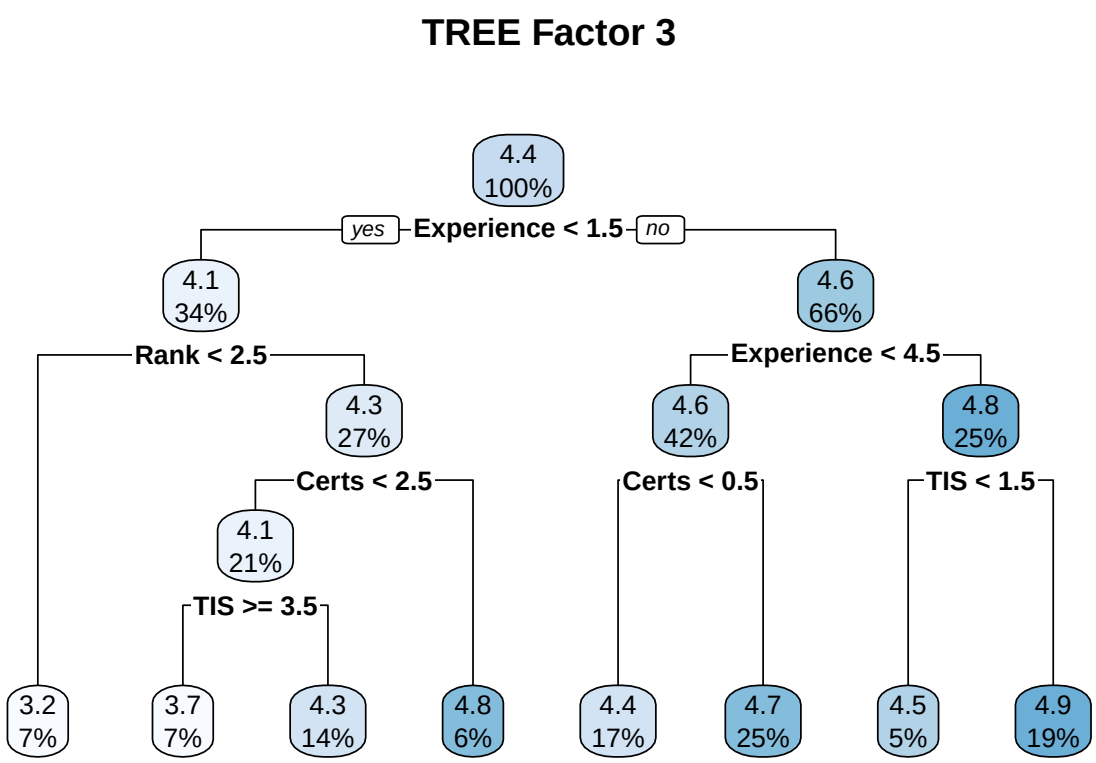

```
# text(treeFact2)

treeFact3<-rpart(FactorScore3~Rank+Age+TIS+BS+Certs+Experience+MS+OPExperience, cp=0.006)
printcp(treeFact3)
```

```
##
## Regression tree:
## rpart(formula = FactorScore3 ~ Rank + Age + TIS + BS + Certs +
##     Experience + MS + OPExperience, cp = 0.006)
##
## Variables actually used in tree construction:
## [1] Certs      Experience Rank       TIS
##
## Root node error: 77.213/134 = 0.57622
##
## n= 134
```

```
##
##          CP nsplit rel error  xerror    xstd
## 1 0.1319868      0   1.00000 1.01957 0.36495
## 2 0.1015174      1   0.86801 1.04588 0.34135
## 3 0.0313155      2   0.76650 1.06879 0.37665
## 4 0.0277206      3   0.73518 1.05419 0.37626
## 5 0.0146558      4   0.70746 0.97559 0.33947
## 6 0.0133507      5   0.69280 0.98246 0.33928
## 7 0.0095105      6   0.67945 0.96975 0.33940
## 8 0.0060000      7   0.66994 0.95753 0.33977
```

```
rpart.plot(treeFact3, main='TREE Factor 3')
```




```
# text(treeFact3)
```

I like this better. Now, we see how good our model is! Linear model is not working great for this.

Now let's take a look at how well this model did with the predicions using the test data, as opposed to the training data.

First Model (Factor 1):

```
resultsdfFact1<-data.frame(
  Actual=testdata1$FactorScore1,
  Prediction=predict(treeFact1, newdata=testdata1)
)
resultsdfFact1$diff<-resultsdfFact1$Actual-resultsdfFact1$Prediction

##("Standard Deviation of the difference between predicted and actual")
```

```r
sd(resultsdfFact1$diff)
```

```
## [1] 1.399393
```

```r
##("Mean of the difference between predicted and actual")
mean(resultsdfFact1$diff)
```

```
## [1] -0.2620682
```

```r
##("Median of the difference between predicted and actual")
median(resultsdfFact1$diff)
```

```
## [1] 0.1728327
```

Second Model (Factor 2):

```r
resultsdfFact2<-data.frame(
  Actual=testdata1$FactorScore2,
  Prediction=predict(treeFact2, newdata=testdata1)
)
resultsdfFact2$diff<-resultsdfFact2$Actual-resultsdfFact2$Prediction

##("Standard Deviation of the difference between predicted and actual")
sd(resultsdfFact2$diff)
```

```
## [1] 1.08705
```

```r
##("Mean of the difference between predicted and actual")
mean(resultsdfFact2$diff)
```

```
## [1] -0.1821727
```

```r
##("Median of the difference between predicted and actual")
median(resultsdfFact2$diff)
```

```
## [1] 0.1613028
```

Third Model (Factor 3):

```r
resultsdfFact3<-data.frame(
  Actual=testdata1$FactorScore3,
  Prediction=predict(treeFact3, newdata=testdata1)
)
resultsdfFact3$diff<-resultsdfFact3$Actual-resultsdfFact3$Prediction

##("Standard Deviation of the difference between predicted and actual")
sd(resultsdfFact3$diff)
```

```
## [1] 1.171089
```

```r
##("Mean of the difference between predicted and actual")
mean(resultsdfFact3$diff)
```

```
## [1] -0.1375787
```

```r
##("Median of the difference between predicted and actual")
median(resultsdfFact3$diff)
```

```
## [1] 0.1100923
```

Final thoughs: the models don't seem too bad, but I wouldn't promote/demote anyone based on the results. They could be a good start, but the trees are a little overfit (not something that you have to meantion, but

you could say that more data is needed for a better model).